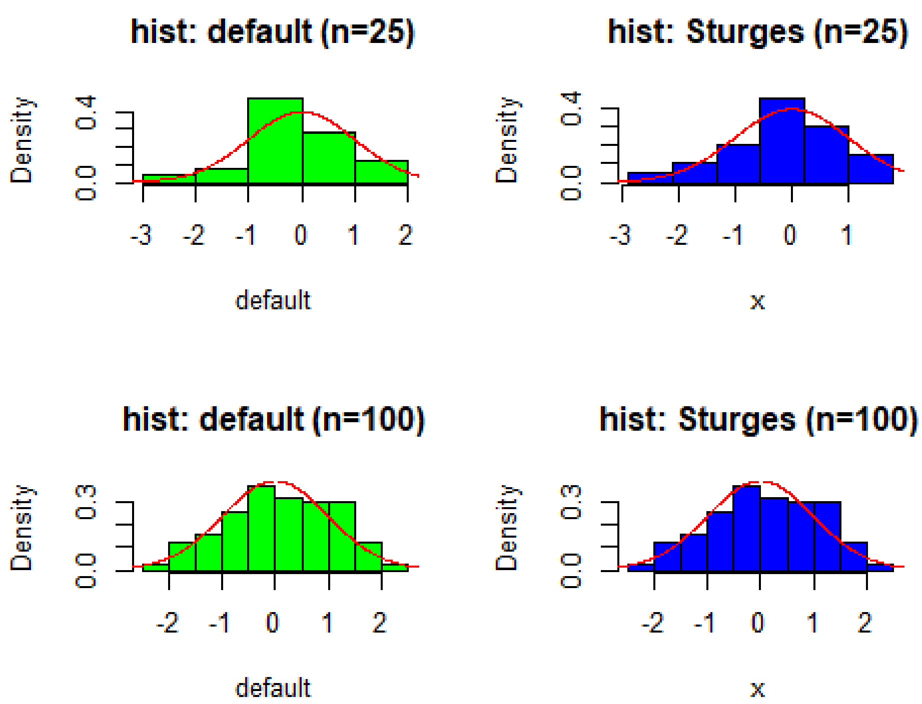
Density-Estimation-s15680.R

2024-03-04

```
## Density Estimation
##s15680
## Codes of the Use cases and examples for the report .
##############################################################################
##################################
#2.Univariate Density Estimations
#2.1 Histograms
## Example_Sturges’rule:
set.seed(20)
par(mfrow = c(2, 2))
n = 25
x = rnorm(n)
# calc breaks according to Sturges’ Rule
nclass = ceiling(1 + log2(n))
cwidth = diff(range(x) / nclass)
breaks = min(x) + cwidth * 0:nclass
h.default = hist(x, freq = FALSE, xlab = "default", main = "hist: default
(n=25)", col = "green")
z = qnorm(ppoints(1000))
lines(z, dnorm(z), col = "red")
h.sturges = hist(x, breaks = breaks, freq = FALSE, main = "hist: Sturges
(n=25)", col = "blue")
lines(z, dnorm(z), col = "red")

n = 100
x = rnorm(n)
# calc breaks according to Sturges’ Rule
nclass = ceiling(1 + log2(n))
cwidth = diff(range(x) / nclass)
breaks = min(x) + cwidth * 0:nclass
h.default = hist(x, freq = FALSE, xlab = "default", main = "hist: default
(n=100)", col = "green")
z = qnorm(ppoints(1000))
lines(z, dnorm(z), col = "red")
h.sturges = hist(x, breaks = "Sturges", freq = FALSE, main = "hist: Sturges
(n=100)", col = "blue")
lines(z, dnorm(z), col = "red")
```

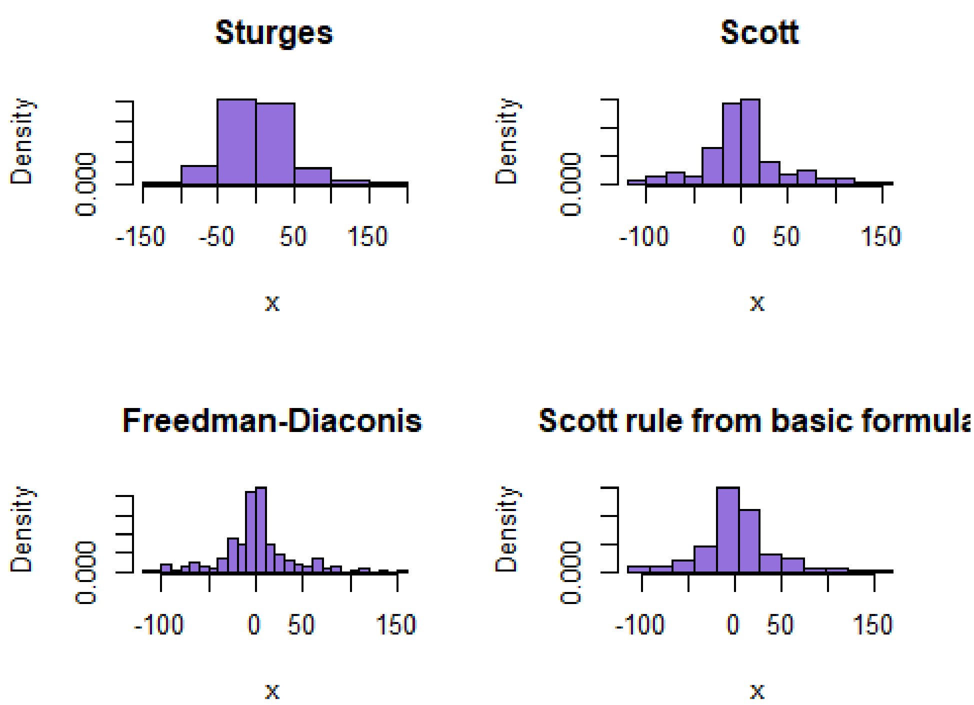



```
print(cwidth)

## [1] 0.585385

nclass.Sturges

## function (x)
## ceiling(log2(length(x)) + 1)
## <bytecode: 0x00000276e9f68ed8>
## <environment: namespace:grDevices>

function (x) ceiling(log2(length(x)) + 1)

## function (x) ceiling(log2(length(x)) + 1)

par(mfrow = c(1, 1))


## Example_Density_estimation_for_Old_Faithful
par(mfrow = c(1, 2))
library(MASS) #for geyser and truehist
waiting = geyser$waiting
n = length(waiting)
# rounding the constant in Scott's rule
# and using sample standard deviation to estimate sigma
h = 3.5 * sd(waiting) * n^(-1/3)
# number of classes is determined by the range and h
m = min(waiting)
```

```r
M = max(waiting)
nclass = ceiling((M - m) / h)
breaks = m + h * 0:nclass

h.scott = hist(waiting, breaks = breaks, freq = FALSE,
                main = "Scott’s Rule")
truehist(waiting, nbins = "Scott", x0 = 0, prob=TRUE,
         col = 0,main = "hist with breaks = 'scott'")

hist(waiting, breaks = "scott", prob=TRUE, density=5,
     add=TRUE)
```

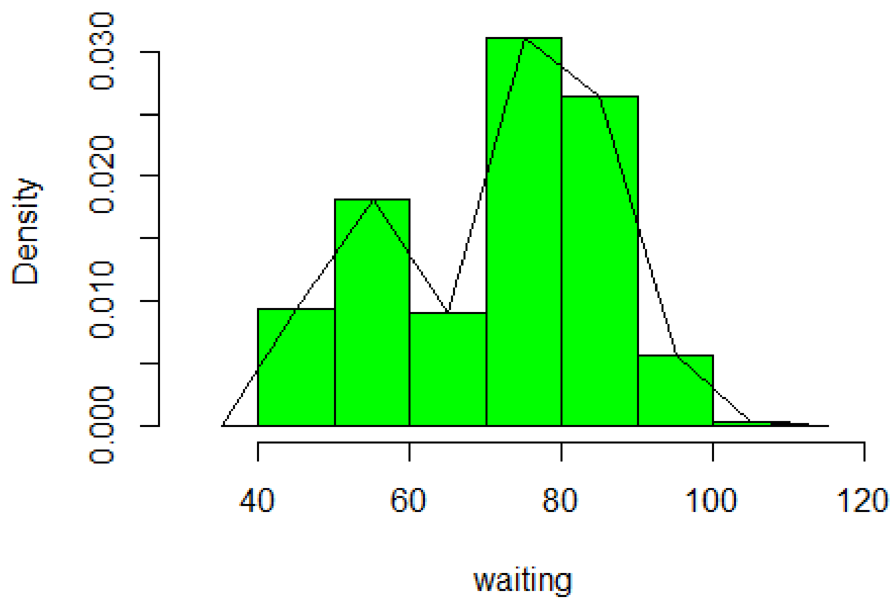

```r
par(mfrow = c(1, 1))


##bin width comparisons
set.seed(100)
n = 200
sig = sample(c(10, 60), n, replace=T)
x = rnorm(n, 0, sig)

# rounding the constant in Scott’s rule
# and using sample standard deviation to estimate sigma
h = 3.5 * sd(x) * n^(-1/3)
# number of classes is determined by the range and h
m = min(x)
```

```
M = max(x)
nclass = ceiling((M - m) / h)
scott.breaks = m + h * 0:nclass
par(mfrow=c(2,2))
hist(x, breaks = "sturges", freq = FALSE, main = "Sturges", col = "#9370DB")
hist(x, breaks = "scott", freq = FALSE,main = "Scott", col = "#9370DB")
hist(x, breaks = "FD", freq = FALSE,
     main = "Freedman-Diaconis", col = "#9370DB")
hist(x, breaks = scott.breaks, freq = FALSE,
     main = "Scott rule from basic formula", col = "#9370DB")
```



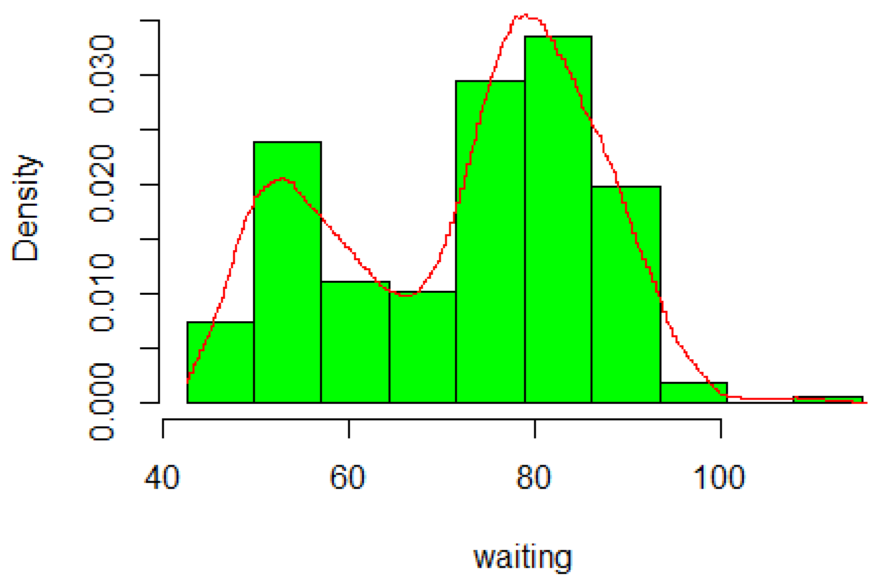

```
par(mfrow=c(1,1))



## 2.2 Frequency polygon density estimate
##  Frequency polygon density estimate)
set.seed(100)
library(MASS) #for geyser
waiting = geyser$waiting #in MASS
n = length(waiting)
# freq poly bin width using normal ref rule
h = 2.15 * sqrt(var(waiting)) * n^(-1/5)
# calculate the sequence of breaks and histogram
br = pretty(waiting, diff(range(waiting)) / h)
brplus = c(min(br)-h, max(br+h))
```

```r
histg = hist(waiting, breaks = br, freq = FALSE,
             main = "", xlim = brplus,col="green")
vx = histg$mids #density est at vertices of polygon
vy = histg$density
delta = diff(vx)[1] # h after pretty is applied
k = length(vx)
vx = vx + delta # the bins on the ends
vx = c(vx[1] - 2 * delta, vx[1] - delta, vx)
vy = c(0, vy, 0)
# add the polygon to the histogram
polygon(vx, vy)
```



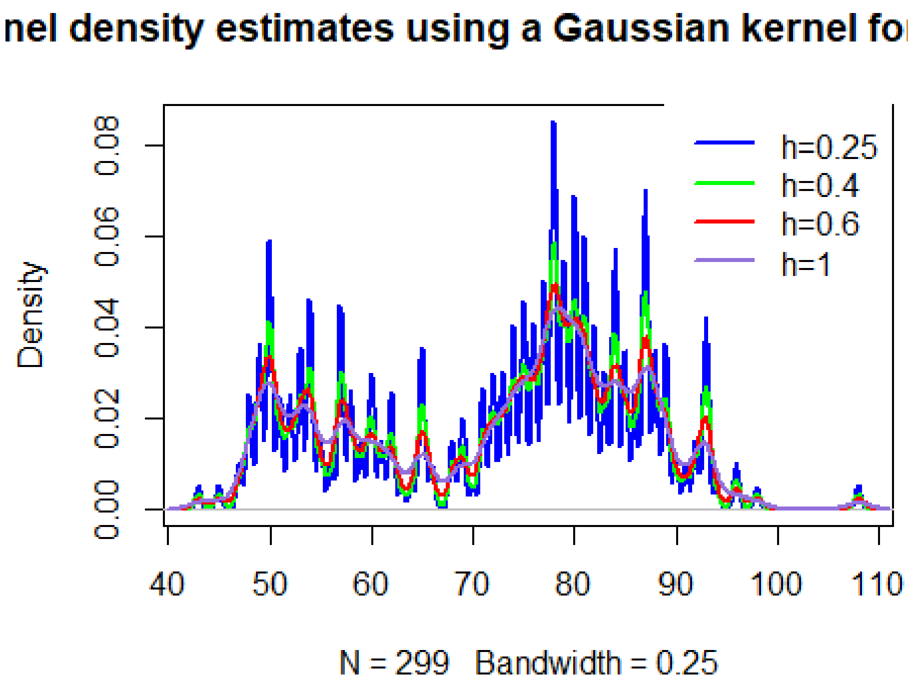

```r
## (ASH density estimate)
## The average shifted histogram
set.seed(100)
library(MASS)
waiting = geyser$waiting
n = length(waiting)
m = 20
a = min(waiting) - .5
b = max(waiting) + .5

h = 7.27037
delta = h / m
#get the bin counts on the delta-width mesh.
br = seq(a - delta*m, b + 2*delta*m, delta)
histg = hist(waiting, breaks = br, plot = FALSE)
```

```r
nk = histg$counts
K = abs((1-m):(m-1))
fhat = function(x) {
  # locate the leftmost interval containing x
  i = max(which(x > br))
  k = (i - m + 1):(i + m - 1)
  # get the 2m-1 bin counts centered at x
  vk = nk[k]
  sum((1 - K / m) * vk) / (n * h) #f.hat
}
# density can be computed at any points in range of data
z = as.matrix(seq(a, b + h, .1))
f.ash = apply(z, 1, fhat) #density estimates at midpts
# plot ASH density estimate over histogram
br2 = seq(a, b + h, h)
hist(waiting, breaks = br2, freq = FALSE, main = "",
     ylim = c(0, max(f.ash)),col="green")
lines(z, f.ash, xlab = "waiting",col="red")
```

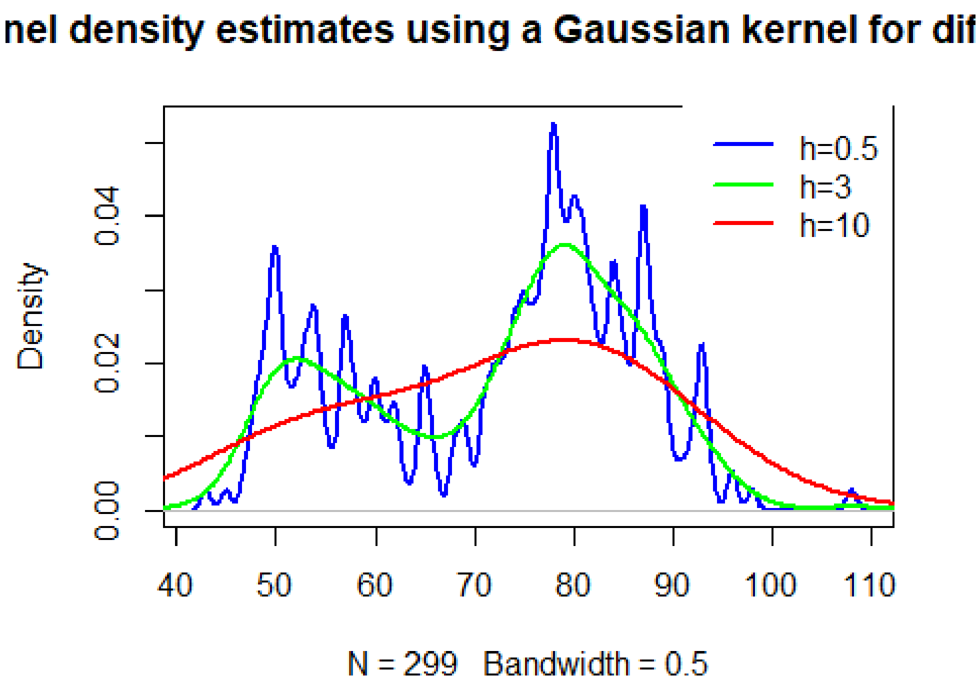



```r
##2.4 Kernel Density Estimation

##Kernel density estimates using a Gaussian kernel with different band widths


#density estimators for different h=0.25,0.4,0.6,1
```

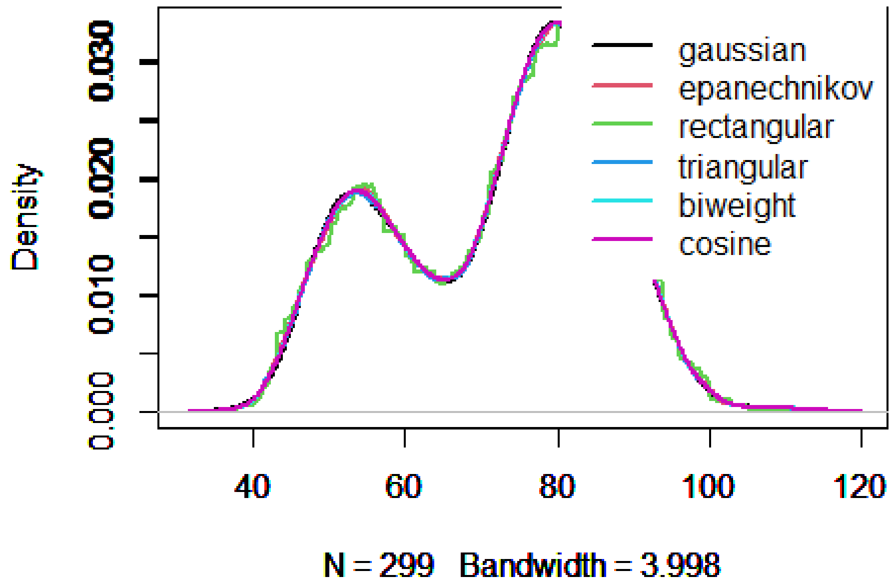
```r
plot(density(waiting,bw =0.25, kernel = "gaussian"),col ='blue', lwd=2,main =
"Kernel density estimates using a Gaussian kernel for different h")
lines(density(waiting,bw =0.4, kernel = "gaussian"),col ='green', lwd=2)
lines(density(waiting,bw =0.6, kernel = "gaussian"),col ='red', lwd=2)
lines(density(waiting,bw =1, kernel = "gaussian"),col ="#9370DB", lwd=2)
legend("topright", c("h=0.25","h=0.4","h=0.6","h=1"), box.lty = 0, col =
c('blue','green','red',"#9370DB"), lwd = c(2, 2, 2,2))
```



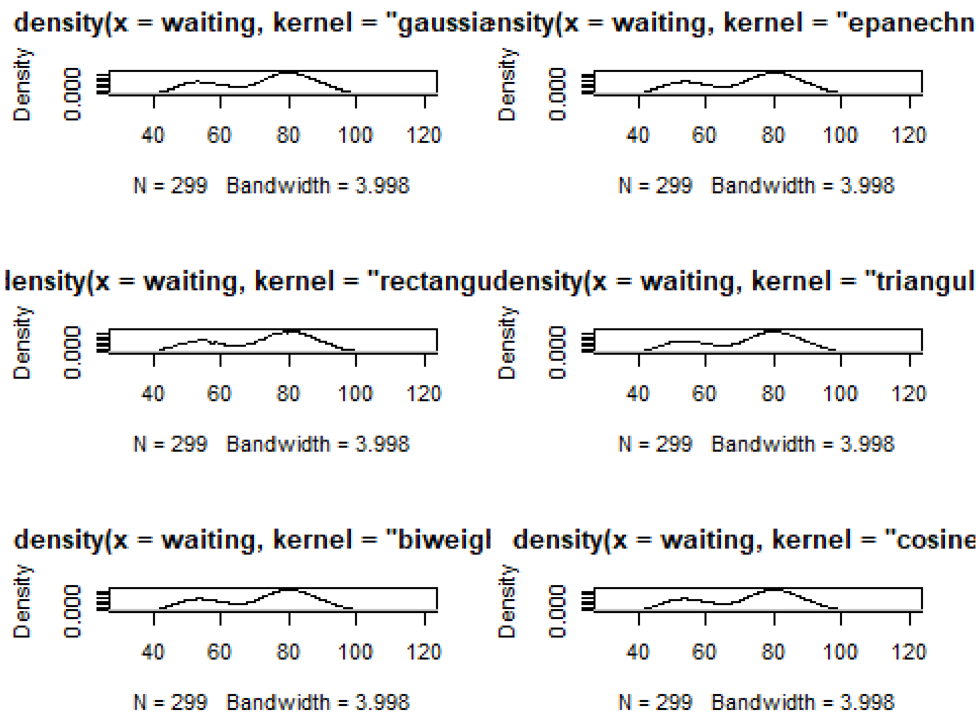

```r
#density estimators for different h=0.5,3,10,

plot(density(waiting,bw =0.5, kernel = "gaussian"),col ='blue', lwd=2,main =
"Kernel density estimates using a Gaussian kernel for different h")
lines(density(waiting,bw =3, kernel = "gaussian"),col ='green', lwd=2)
lines(density(waiting,bw =10, kernel = "gaussian"),col ='red', lwd=2)
legend("topright", c("h=0.5","h=3","h=10"), box.lty = 0, col =
c('blue','green','red'), lwd = c(2, 2, 2))
```



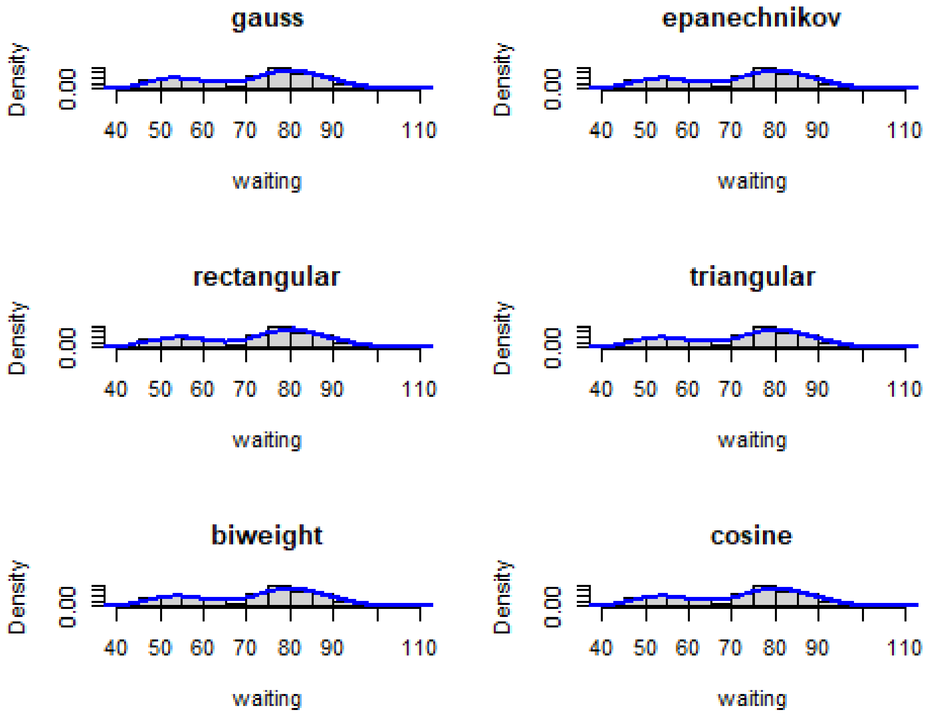

```r
##Kernel Functions that are Commonly Applied in Density Estimation
#density estimators for different kernel functions
waiting = geyser$waiting
par(mfrow=c(1,1))
#different kernel functions
plot(density(waiting,kernel = "gaussian"), lwd = 2, col = 1,main="Different
Kernel functions")
par(new=T)
plot(density(waiting,kernel = "epanechnikov"), lwd = 2, col = 2)
par(new=T)
plot(density(waiting,kernel = "rectangular"), lwd = 2, col = 3)
par(new=T)
plot(density(waiting,kernel = "triangular"), lwd = 2, col = 4)
par(new=T)
plot(density(waiting,kernel = "biweight"), lwd = 2, col = 5)
par(new=T)
plot(density(waiting,kernel = "cosine"), lwd = 2, col = 6)
par(new=F)
legend("topright",
c("gaussian","epanechnikov","rectangular","triangular","biweight","cosine"),
box.lty = 0, col = c(1,2,3,4,5,6), lwd = c(2, 2, 2))
```



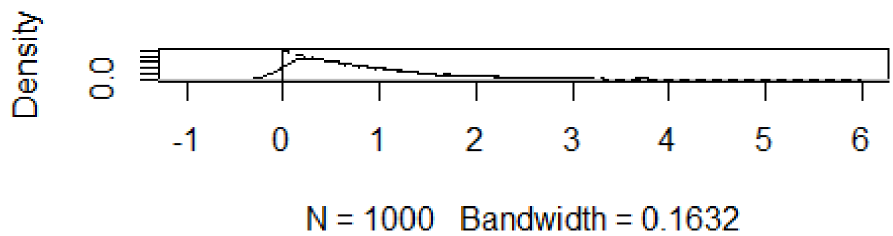

```r
#Different kernel functions that are commonly applied in Density Estimations
using waiting
par(mfrow=c(3,2))
plot(density(waiting,kernel = "gaussian"))
plot(density(waiting,kernel = "epanechnikov"))
plot(density(waiting,kernel = "rectangular"))
plot(density(waiting,kernel = "triangular"))
plot(density(waiting,kernel = "biweight"))
plot(density(waiting,kernel = "cosine"))
```

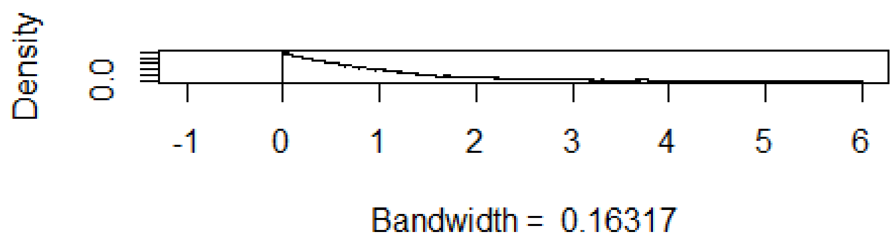



```r
#density estimators for different kernel functions
par(mfrow=c(3,2))
hist(waiting,freq = F ,main="gauss")#histogram
lines(density(waiting,kernel = "gaussian"), lwd = 2, col = 'blue')
hist(waiting,freq = F,main="epanechnikov" )#histogram
lines(density(waiting,kernel = "epanechnikov"), lwd = 2, col = 'blue')
hist(waiting,freq = F,main="rectangular" )#histogram
lines(density(waiting,kernel = "rectangular"), lwd = 2, col = 'blue')
hist(waiting,freq = F ,main="triangular")#histogram
lines(density(waiting,kernel = "triangular"), lwd = 2, col = 'blue')
hist(waiting,freq = F,main="biweight" )#histogram
lines(density(waiting,kernel = "biweight"), lwd = 2, col = 'blue')
hist(waiting,freq = F,main="cosine")#histogram
lines(density(waiting,kernel = "cosine"), lwd = 2, col = 'blue')
```

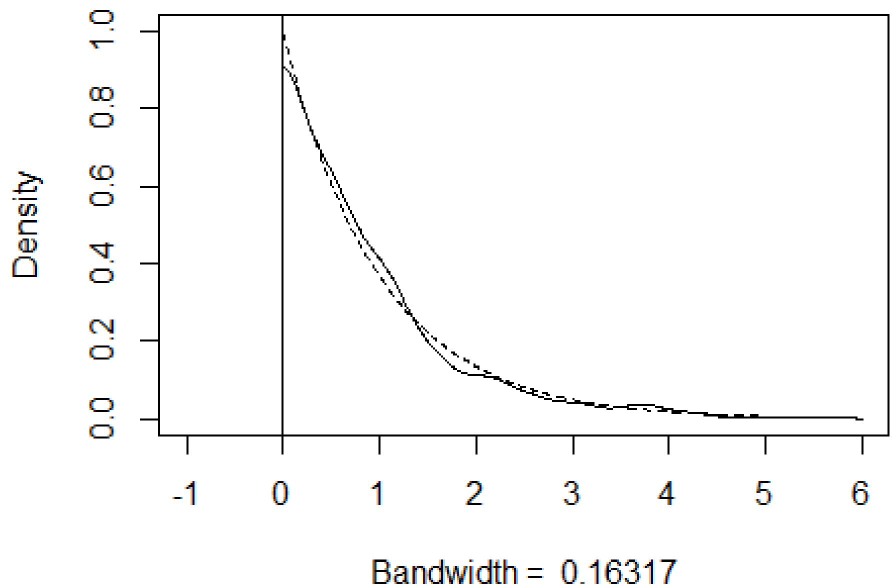



```r
par(mfrow=c(1,1))

## 2.5 Boundary kernels
##Example_(Exponential density):

set.seed(100)
par(mfrow=c(2,1))
x = rexp(1000, 1)
plot(density(x), xlim = c(-1, 6), ylim = c(0, 1), main="")
abline(v = 0)
# add the true density to compare
y = seq(.001, 6, .01)
lines(y, dexp(y, 1), lty = 2)

xx = c(x, -x)
g = density(xx, bw = bw.nrd0(x))
a = seq(0, 6, .01)
ghat = approx(g$x, g$y, xout = a)
fhat = 2 * ghat$y # density estimate along a
bw = paste("Bandwidth = ", round(g$bw, 5))
plot(a, fhat, type="l", xlim=c(-1, 6), ylim=c(0, 1),
     main = "", xlab = bw, ylab = "Density")
abline(v = 0)
# add the true density to compare
y = seq(.001, 6, .01)
lines(y, dexp(y, 1), lty = 2)
```





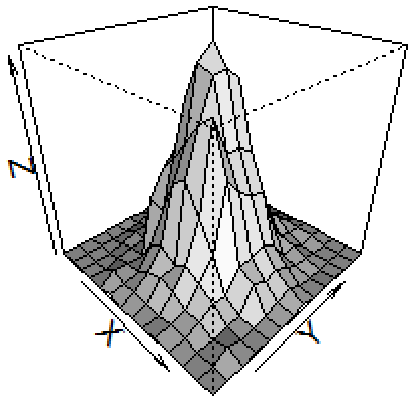

```r
par(mfrow=c(1,1))


## 2.5.1 Reflection Boundary Technique
set.seed(100)
xx = c(x, -x)
g = density(xx, bw = bw.nrd0(x))
a = seq(0, 6, .01)
ghat = approx(g$x, g$y, xout = a)
fhat = 2 * ghat$y # density estimate along a
bw = paste("Bandwidth = ", round(g$bw, 5))
plot(a, fhat, type="l", xlim=c(-1, 6), ylim=c(0, 1),
     main = "", xlab = bw, ylab = "Density")
abline(v = 0)
# add the true density to compare
y = seq(.001, 6, .01)
lines(y, dexp(y, 1), lty = 2)
```




```
####################################################################################################################
### 3 Bivariate and Multivariate Density Estimations

## 3.1 Bivariate Frequency Polygon
## (Bivariate frequency table: bin2d)


bin2d =
  function(x, breaks1 = "Sturges", breaks2 = "Sturges"){
    # Data matrix x is n by 2
    # breaks1, breaks2: any valid breaks for hist function
    # using same defaults as hist
    histg1 = hist(x[,1], breaks = breaks1, plot = FALSE)
    histg2 = hist(x[,2], breaks = breaks2, plot = FALSE)
    brx = histg1$breaks
    bry = histg2$breaks
    # bin frequencies
    freq = table(cut(x[,1], brx), cut(x[,2], bry))
    return(list(call = match.call(), freq = freq,
                breaks1 = brx, breaks2 = bry,
                mids1 = histg1$mids, mids2 = histg2$mids))
  }

bin2d(iris[1:50,1:2])
```

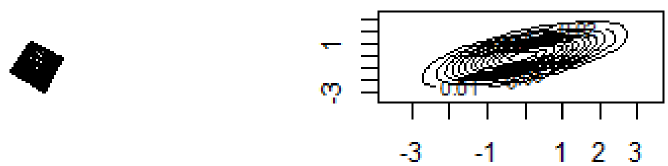
```
## $call
## bin2d(x = iris[1:50, 1:2])
##
## $freq
##
##            (2,2.5] (2.5,3] (3,3.5] (3.5,4] (4,4.5]
##   (4.2,4.4]       0       3       1       0       0
##   (4.4,4.6]       1       0       3       1       0
##   (4.6,4.8]       0       2       5       0       0
##   (4.8,5]         0       2       8       2       0
##   (5,5.2]         0       0       6       4       1
##   (5.2,5.4]       0       0       2       4       0
##   (5.4,5.6]       0       0       1       0       1
##   (5.6,5.8]       0       0       0       2       1
##
## $breaks1
## [1] 4.2 4.4 4.6 4.8 5.0 5.2 5.4 5.6 5.8
##
## $breaks2
## [1] 2.0 2.5 3.0 3.5 4.0 4.5
##
## $mids1
## [1] 4.3 4.5 4.7 4.9 5.1 5.3 5.5 5.7
##
## $mids2
## [1] 2.25 2.75 3.25 3.75 4.25
```

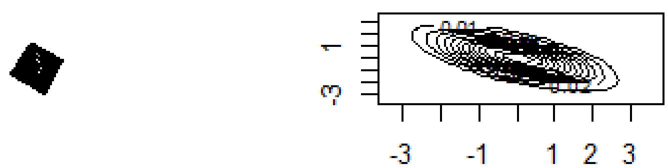
```
## Example_(Bivariate_density_polygon)


#generate standard bivariate normal random sample
n = 2000; d = 2
x = matrix(rnorm(n*d), n, d)
# compute the frequency table and density estimates
b = bin2d(x)
h1 = diff(b$breaks1)
h2 = diff(b$breaks2)
# matrix h contains the areas of the bins in b
h = outer(h1, h2, "*")
Z = b$freq / (n * h) # the density estimate
persp(x=b$mids1, y=b$mids2, z=Z, shade=TRUE,
      xlab="X", ylab="Y", main="",
      theta=45, phi=30, ltheta=60)
```



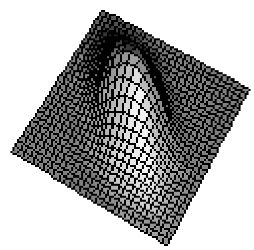

```r
## (Bivariate ASH library(ash)
## Example_(Bivariate_ASH_density_estimate)
library(ash) # for bivariate ASH density est.
# generate N_2(0,Sigma) data
n = 2000
d = 2
nbin = c(30, 30) # number of bins
m = c(5, 5) # smoothing parameters
# First example with positive correlation
Sigma = matrix(c(1, .9, .9, 1), 2, 2)
set.seed(345)
library(MASS) #for mvrnorm()
x = mvrnorm(n, c(0, 0), Sigma=Sigma)
b = bin2(x, nbin = nbin)
# kopt is the kernel type, here triangular
est = ash2(b, m = m, kopt = c(1,0))
par(mfrow= c(2,2))
persp(x = est$x, y = est$y, z = est$z, shade=TRUE,
      xlab = "X", ylab = "Y", zlab = "", main="",
      theta = 30, phi = 75, ltheta = 30, box = FALSE)
contour(x = est$x, y = est$y, z = est$z, main="")

# Second example with negative correlation
Sigma = matrix(c(1, -.9, -.9, 1), 2, 2)
set.seed(345)
#x = rmvn.eigen(n, c(0, 0), Sigma=Sigma)
```

```r
x = mvrnorm(n, c(0, 0), Sigma=Sigma)
b = bin2(x, nbin = nbin)
est = ash2(b, m = m, kopt = c(1,0))
persp(x = est$x, y = est$y, z = est$z, shade=TRUE,
      xlab = "X", ylab = "Y", zlab = "", main="",
      theta = 30, phi = 75, ltheta = 30, box = FALSE)
contour(x = est$x, y = est$y, z = est$z, main="")
```

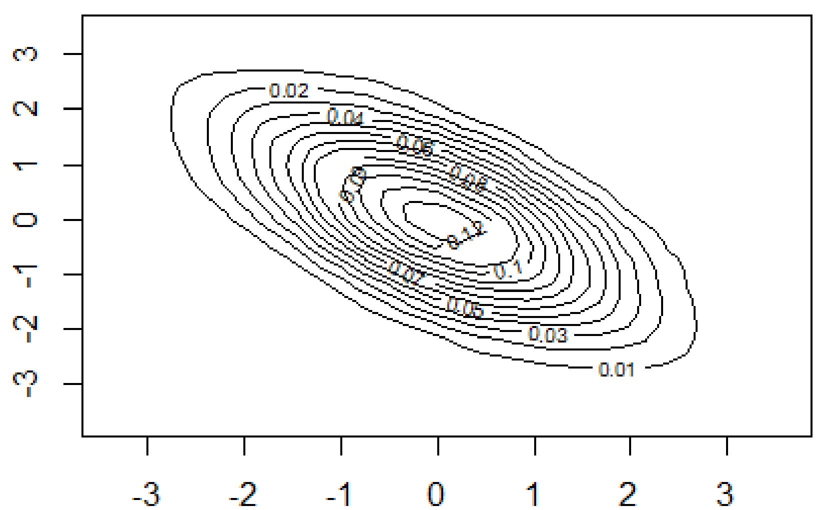




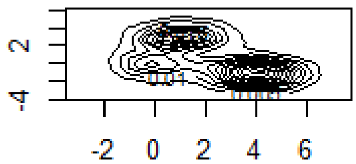
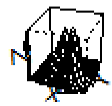
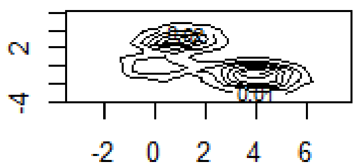
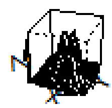

```r
par(mfrow= c(1,1))




# Second example with negative correlation
library(ash)
library(MASS)

Sigma = matrix(c(1, -.9, -.9, 1), 2, 2)
set.seed(345)
x = mvrnorm(n, c(0, 0), Sigma)
b = bin2(x, nbin = nbin)
est = ash2(b, m = m, kopt = c(1,0))
persp(x = est$x, y = est$y, z = est$z, shade=TRUE,
      xlab = "X", ylab = "Y", zlab = "", main="",
      theta = 30, phi = 75, ltheta = 30, box = FALSE)
```



```r
contour(x = est$x, y = est$y, z = est$z, main="")
```



```r
par(ask = FALSE)



#### Multidimensional kernel methods
## Example(Product kernel estimate of a bivariate normal mixture)
par(mfrow= c(2,2))
library(MASS) #for mvrnorm and kde2d
#generate the normal mixture data
n = 2000
p = c(.2, .3, .5)
mu = matrix(c(0, 1, 4, 0, 3, -1), 3, 2)
Sigma = diag(2)
i = sample(1:3, replace = TRUE, prob = p, size = n)
k = table(i)
x1 = mvrnorm(k[1], mu = mu[1,], Sigma)
x2 = mvrnorm(k[2], mu = mu[2,], Sigma)
x3 = mvrnorm(k[3], mu = mu[3,], Sigma)
X = rbind(x1, x2, x3) #the mixture data
x = X[,1]
y = X[,2]
 print(c(bandwidth.nrd(x), bandwidth.nrd(y)))

## [1] 1.842843 1.878773

# accepting the default normal reference bandwidth
fhat = kde2d(x, y)
contour(fhat)
persp(fhat, phi = 30, theta = 20, d = 5, xlab ="x")
# select bandwidth by unbiased cross-validation
h = c(ucv(x), ucv(y))
fhat = kde2d(x, y, h = h)
contour(fhat)
persp(fhat, phi = 30, theta = 20, d = 5, xlab = "x")
```

```
par(mfrow= c(1,1))
```